Faculdade: Estácio De Sá

Polo: Campo Grande

Aluno: Luis Henrique Peçanha Machado

Jogo de nave em Java com POO, Swing e JDBC

Visão:

Main.java

```java
// Main.java
package visao;

import modelo.*;
import banco.PontuacaoDAO;
import modelo.Pontuacao;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Main extends JFrame {
    private String jogadorNome;
    private NaveEspacial nave;
    private final ArrayList<Inimigo> inimigos = new ArrayList<>();
    private final ArrayList<Tiro> tiros = new ArrayList<>();
    private final ArrayList<Tiro> tirosInimigos = new ArrayList<>();
    private ArrayList<Estrela> estrelas = new ArrayList<>();
    private final JPanel painel;
    private JLabel labelPontuacao;
    private boolean jogoAtivo = true;
    private int pontuacao = 0;
    private static final int LIMITE_INIMIGOS = 20;

    public Main() {
        jogadorNome = JOptionPane.showInputDialog("Digite seu nome:");
        setTitle("Jogo de Nave Espacial - Piloto: " + jogadorNome);
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);

        painel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.setColor(Color.BLACK);
                g.fillRect(0, 0, getWidth(), getHeight());

                g.setColor(Color.WHITE);
                for (Estrela estrela : estrelas) {
                    g.fillOval(estrela.getX(), estrela.getY(), 3, 3);
                }
```

```
                if (jogoAtivo && nave != null) {
                    g.setColor(nave.getCor());
                    g.fillPolygon(nave.getXPoints(),
nave.getYPoints(), 3);
                }

                g.setColor(Color.YELLOW);
                for (Tiro tiro : tiros) {
                    g.fillRect(tiro.getPosX(), tiro.getPosY(), 5, 10);
                }

                g.setColor(Color.RED);
                for (Tiro tiroInimigo : tirosInimigos) {
                    g.fillRect(tiroInimigo.getPosX(),
tiroInimigo.getPosY(), 5, 10);
                }

                for (Inimigo inimigo : inimigos) {
                    if (inimigo.getTipo() == 0) {
                        g.setColor(Color.GRAY);
                        g.fillRect(inimigo.getPosX(),
inimigo.getPosY(), 30, 30);
                    } else {
                        g.setColor(Color.RED);
                        int[] xPoints = { inimigo.getPosX(),
inimigo.getPosX() + 15, inimigo.getPosX() + 30 };
                        int[] yPoints = { inimigo.getPosY(),
inimigo.getPosY() + 30, inimigo.getPosY() };
                        g.fillPolygon(xPoints, yPoints, 3);
                    }
                }
            }
        };

        labelPontuacao = new JLabel("Pontuação: 0");
        labelPontuacao.setForeground(Color.WHITE);
        painel.setLayout(new BorderLayout());
        painel.add(labelPontuacao, BorderLayout.NORTH);

        add(painel);
        iniciarNave();
        iniciarAmbiente();
        configurarControles();
        iniciarMovimentacao();
        setVisible(true);
    }

    private void iniciarNave() {
        String escolha = JOptionPane.showInputDialog("Escolha sua
nave:\n1 - Ataque\n2 - Defesa\n3 - Exploradora");
        switch (escolha) {
            case "1" -> nave = new NaveAtaque(375, 500);
            case "2" -> nave = new NaveDefesa(375, 500);
            case "3" -> nave = new NaveExploradora(375, 500);
            default -> nave = new NaveAtaque(375, 500);
        }
    }

    private void iniciarAmbiente() {
        estrelas = new ArrayList<>();
        for (int i = 0; i < 50; i++) {
```

```
            estrelas.add(new Estrela(getWidth()));
        }

        Timer ambienteTimer = new Timer(50, e -> {
            for (Estrela estrela : estrelas) estrela.mover();
            painel.repaint();
        });
        ambienteTimer.start();
    }

    private void configurarControles() {
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (jogoAtivo && nave != null) {
                    switch (e.getKeyCode()) {
                        case KeyEvent.VK_LEFT -> nave.mover('L');
                        case KeyEvent.VK_RIGHT -> nave.mover('R');
                        case KeyEvent.VK_UP -> nave.mover('U');
                        case KeyEvent.VK_DOWN -> nave.mover('D');
                        case KeyEvent.VK_SPACE -> tiros.add(new
Tiro(nave.getPosX() + 22, nave.getPosY()));
                    }
                    repaint();
                }
            }
        });
    }

    private void iniciarMovimentacao() {
        Timer timer = new Timer(50, e -> {
            if (!jogoAtivo) return;

            nave.avancar();

            Iterator<Inimigo> iterInimigos = inimigos.iterator();
            while (iterInimigos.hasNext()) {
                Inimigo inimigo = iterInimigos.next();
                inimigo.mover();
                if (inimigo.getPosY() > getHeight())
iterInimigos.remove();
                else if (inimigo.getTipo() == 1 && Math.random() <
0.04)
                    tirosInimigos.add(new Tiro(inimigo.getPosX() + 12,
inimigo.getPosY() + 30));
            }

            Iterator<Tiro> iterTiros = tiros.iterator();
            while (iterTiros.hasNext()) {
                Tiro tiro = iterTiros.next();
                tiro.mover();
                boolean removeTiro = false;

                Iterator<Inimigo> iterColisao = inimigos.iterator();
                while (iterColisao.hasNext()) {
                    Inimigo inimigo = iterColisao.next();
                    if (Math.abs(tiro.getPosX() - inimigo.getPosX()) <
30 &&
                            Math.abs(tiro.getPosY() -
inimigo.getPosY()) < 30) {
                        iterColisao.remove();
```

```
                        removeTiro = true;
                        pontuacao += 10;
                        break;
                    }
                }

                if (removeTiro || tiro.getPosY() < 0)
iterTiros.remove();
            }

            Iterator<Tiro> iterTirosInimigos =
tirosInimigos.iterator();
            while (iterTirosInimigos.hasNext()) {
                Tiro tiro = iterTirosInimigos.next();
                tiro.moverInverso();
                if (tiro.getPosY() > getHeight())
iterTirosInimigos.remove();
                else if (Math.abs(tiro.getPosX() - nave.getPosX()) <
30 &&
                        Math.abs(tiro.getPosY() - nave.getPosY()) <
30) {
                    jogoAtivo = false;
                    salvarPontuacao();
                    exibirRanking();
                    System.exit(0);
                }
            }

            if (Math.random() < 0.03 && inimigos.size() <
LIMITE_INIMIGOS) {
                inimigos.add(new Inimigo(getWidth()));
            }

            for (Inimigo inimigo : inimigos) {
                if (Math.abs(inimigo.getPosX() - nave.getPosX()) < 30
&&
                        Math.abs(inimigo.getPosY() - nave.getPosY()) <
30) {
                    jogoAtivo = false;
                    salvarPontuacao();
                    exibirRanking();
                    System.exit(0);
                }
            }

            pontuacao++;
            labelPontuacao.setText("Pontuação: " + pontuacao);
            repaint();
        });

        timer.start();
    }

    private void salvarPontuacao() {
        try {
            PontuacaoDAO dao = new PontuacaoDAO();
            Pontuacao registro = new Pontuacao(jogadorNome, pontuacao,
LocalDate.now());
            dao.inserirPontuacao(registro);
        } catch (Exception ex) {
            ex.printStackTrace();
```

```
        }
    }

    private void exibirRanking() {
        try {
            PontuacaoDAO dao = new PontuacaoDAO();
            List<Pontuacao> top5 = dao.buscarTop5Pontuacoes();

            StringBuilder sb = new StringBuilder("🏆 Top 5
Pontuações:\n");
            for (Pontuacao p : top5) {
                sb.append(p.getNome())
                        .append(" - ")
                        .append(p.getPontuacao())
                        .append(" pts (")
                        .append(p.getData())
                        .append(")\n");
            }

            JOptionPane.showMessageDialog(this, sb.toString(),
"Ranking", JOptionPane.INFORMATION_MESSAGE);
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Erro ao exibir o
ranking:\n" + ex.getMessage(),
                    "Erro", JOptionPane.ERROR_MESSAGE);
        }
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

Modelo:

NaveEspacial.java

```
package modelo;

import java.awt.*;

public abstract class NaveEspacial {
    protected int posX, posY;
    protected int velocidade;
    protected final int deslocamentoAutomatico = 1;

    public NaveEspacial(int posX, int posY, int velocidade) {
        this.posX = posX;
        this.posY = posY;
        this.velocidade = velocidade;
    }

    public void mover(char direcao) {
        switch (direcao) {
            case 'L' -> posX -= velocidade;
            case 'R' -> posX += velocidade;
            case 'U' -> posY -= velocidade;
            case 'D' -> posY += velocidade;
        }
```

```
    }

    public void avancar() {
        posY -= deslocamentoAutomatico;
    }

    // Criando métodos getter para acessar corretamente as posições
    public int getPosX() {
        return posX;
    }

    public int getPosY() {
        return posY;
    }

    public abstract Color getCor();
    public abstract int[] getXPoints();
    public abstract int[] getYPoints();
}
```

NaveAtaque.java

```
package modelo;

import java.awt.Color;

public class NaveAtaque extends NaveEspacial {
    public NaveAtaque(int posX, int posY) {
        super(posX, posY, 5);
    }

    public Color getCor() {
        return Color.WHITE;
    }

    public int[] getXPoints() {
        return new int[]{posX, posX + 25, posX + 50};
    }

    public int[] getYPoints() {
        return new int[]{posY + 50, posY, posY + 50};
    }
}
```

NaveDefesa.java

```
package modelo;

import java.awt.Color;

public class NaveDefesa extends NaveEspacial {
    public NaveDefesa(int posX, int posY) {
        super(posX, posY, 3);
    }

    public Color getCor() {
```

```
        return Color.BLUE;
    }

    public int[] getXPoints() {
        return new int[]{posX, posX + 30, posX + 60};
    }

    public int[] getYPoints() {
        return new int[]{posY + 50, posY, posY + 50};
    }
}
```

NaveExploradora.java

```
package modelo;

import java.awt.Color;

public class NaveExploradora extends NaveEspacial {
    public NaveExploradora(int posX, int posY) {
        super(posX, posY, 4); // Nave mais ágil que as outras
    }

    public Color getCor() {
        return Color.GREEN;
    }

    public int[] getXPoints() {
        return new int[]{posX, posX + 20, posX + 40};
    }

    public int[] getYPoints() {
        return new int[]{posY + 50, posY, posY + 50};
    }
}
```

Inimigo.java

```
package modelo;

import java.util.Random;

public class Inimigo {
    private int posX, posY, velocidade;
    private final int tipo; // 0 = meteoro, 1 = nave inimiga

    public Inimigo(int larguraTela) {
        Random random = new Random();
        this.tipo = random.nextInt(2);
        this.posX = random.nextInt(larguraTela - 50);
        this.posY = 0;
        this.velocidade = tipo == 0 ? 3 : 4;
    }

    public void mover() {
        posY += velocidade;
    }
```

```
    public int getPosX() { return posX; }
    public int getPosY() { return posY; }
    public int getTipo() { return tipo; }
}
```

Tiro.java

```
// Tiro.java
package modelo;

public class Tiro {
    private int posX, posY;
    private final int velocidade = 7;

    public Tiro(int posX, int posY) {
        this.posX = posX;
        this.posY = posY;
    }

    public void mover() {
        posY -= velocidade;
    }

    // Método usado para tiros inimigos (movimento para baixo)
    public void moverInverso() {
        posY += velocidade;
    }

    public int getPosX() { return posX; }
    public int getPosY() { return posY; }
}
```

Estrela.java

```
package modelo;

import java.util.Random;

public class Estrela {
    private int x, y, velocidade;

    public Estrela(int larguraTela) {
        Random random = new Random();
        this.x = random.nextInt(larguraTela);
        this.y = random.nextInt(600);
        this.velocidade = 2 + random.nextInt(3);
    }

    public void mover() {
        y += velocidade;
        if (y > 600) y = 0;
    }

    public int getX() { return x; }
    public int getY() { return y; }
}
```

Controle:

Pontuacao.java

```java
package modelo;

import java.time.LocalDate;

public class Pontuacao {
    private String nome;
    private int pontuacao;
    private LocalDate data;

    public Pontuacao(String nome, int pontuacao, LocalDate data) {
        this.nome = nome;
        this.pontuacao = pontuacao;
        this.data = data;
    }

    public String getNome() {
        return nome;
    }

    public int getPontuacao() {
        return pontuacao;
    }

    public LocalDate getData() {
        return data;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void setPontuacao(int pontuacao) {
        this.pontuacao = pontuacao;
    }

    public void setData(LocalDate data) {
        this.data = data;
    }
}
```

Banco:

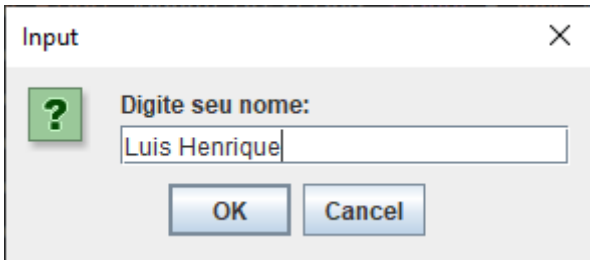
Conexao.java

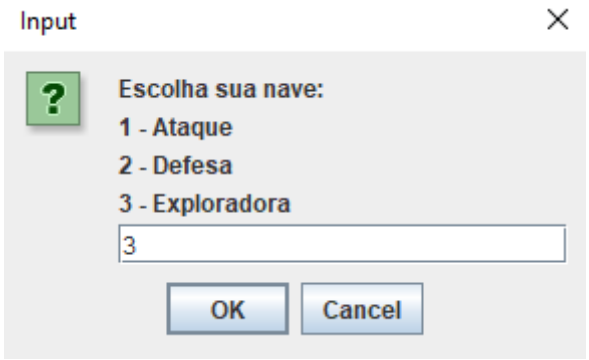
```java
package banco;

import java.sql.Connection;
import java.sql.DriverManager;

public class Conexao {
    private static final String URL =
"jdbc:sqlite:banco_pontuacoes.db";

    public static Connection conectar() throws Exception {
        return DriverManager.getConnection(URL);
    }
}
```

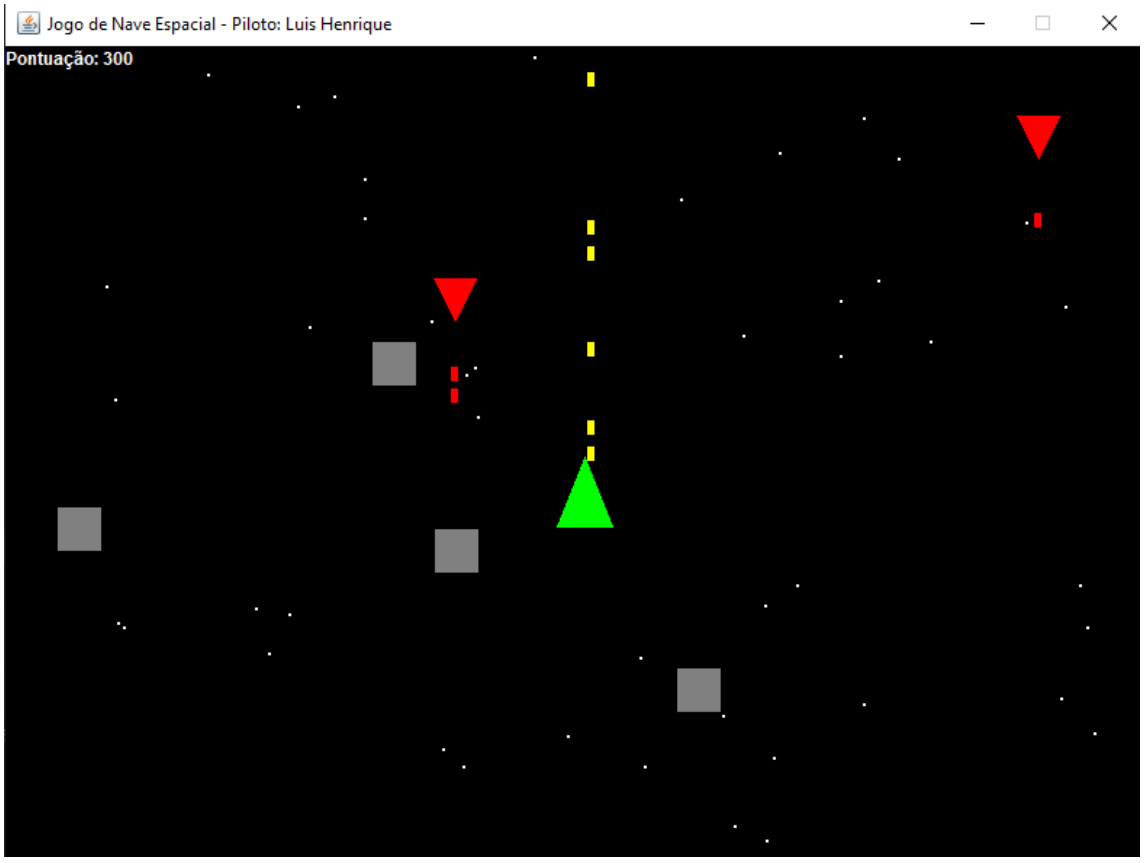
PontuacaoDAO.java

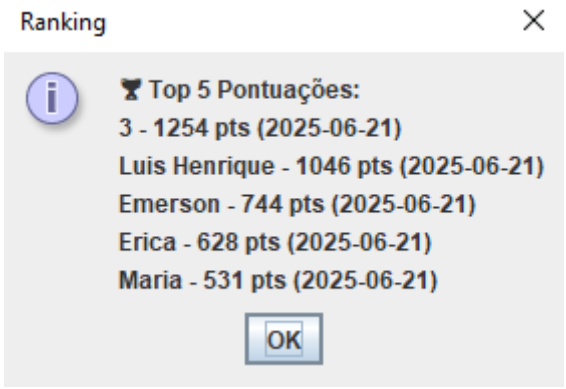
```
package banco;

import modelo.Pontuacao;

import java.sql.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class PontuacaoDAO {

    public PontuacaoDAO() throws Exception {
        criarTabelaSeNaoExistir();
    }

    private void criarTabelaSeNaoExistir() throws Exception {
        String sql = """
            CREATE TABLE IF NOT EXISTS pontuacoes (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nome_jogador TEXT NOT NULL,
                pontuacao INTEGER NOT NULL,
                data DATE NOT NULL
            );
        """;

        try (Connection conn = banco.Conexao.conectar();
PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.execute();
        }
    }

    public void inserirPontuacao(Pontuacao p) throws Exception {
        String sql = "INSERT INTO pontuacoes (nome_jogador, pontuacao,
data) VALUES (?, ?, ?)";

        try (Connection conn = banco.Conexao.conectar();
PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, p.getNome());
            stmt.setInt(2, p.getPontuacao());
            stmt.setDate(3, Date.valueOf(p.getData()));
            stmt.executeUpdate();
        }
    }

    public List<Pontuacao> buscarTop5Pontuacoes() throws Exception {
        String sql = "SELECT nome_jogador, pontuacao, data FROM
pontuacoes ORDER BY pontuacao DESC LIMIT 5";
        List<Pontuacao> top5 = new ArrayList<>();

        try (Connection conn = banco.Conexao.conectar();
PreparedStatement stmt = conn.prepareStatement(sql); ResultSet rs =
stmt.executeQuery()) {
            while (rs.next()) {
                String nome = rs.getString("nome_jogador");
                int pontos = rs.getInt("pontuacao");
                LocalDate data = rs.getDate("data").toLocalDate();
                top5.add(new Pontuacao(nome, pontos, data));
            }
        }
```

```
        return top5;
    }
}
```


Input
Digite seu nome:
Luis Henrique
OK
Cancel


Input
Escolha sua nave:
1 - Ataque
2 - Defesa
3 - Exploradora
3
OK
Cancel


Jogo de Nave Espacial - Piloto: Luis Henrique
Pontuação: 300


Ranking
Top 5 Pontuações:
3 - 1254 pts (2025-06-21)
Luis Henrique - 1046 pts (2025-06-21)
Emerson - 744 pts (2025-06-21)
Erica - 628 pts (2025-06-21)
Maria - 531 pts (2025-06-21)
OK